

**The Atom Syndication Format**  
**draft-ietf-atompub-format-11**

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on February 16, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document specifies Atom, an XML-based Web content and metadata syndication format.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">1.1</a>	<a href="#">Examples . . . . .</a>	<a href="#">3</a>
<a href="#">1.2</a>	<a href="#">Namespace and Version . . . . .</a>	<a href="#">5</a>
<a href="#">1.3</a>	<a href="#">Notational Conventions . . . . .</a>	<a href="#">5</a>
<a href="#">2.</a>	<a href="#">Atom Documents . . . . .</a>	<a href="#">6</a>
<a href="#">3.</a>	<a href="#">Common Atom Constructs . . . . .</a>	<a href="#">8</a>
<a href="#">3.1</a>	<a href="#">Text Constructs . . . . .</a>	<a href="#">8</a>
<a href="#">3.1.1</a>	<a href="#">The "type" Attribute . . . . .</a>	<a href="#">8</a>
<a href="#">3.2</a>	<a href="#">Person Constructs . . . . .</a>	<a href="#">11</a>
<a href="#">3.2.1</a>	<a href="#">The "atom:name" Element . . . . .</a>	<a href="#">11</a>
<a href="#">3.2.2</a>	<a href="#">The "atom:uri" Element . . . . .</a>	<a href="#">11</a>
<a href="#">3.2.3</a>	<a href="#">The "atom:email" Element . . . . .</a>	<a href="#">11</a>
<a href="#">3.3</a>	<a href="#">Date Constructs . . . . .</a>	<a href="#">12</a>
<a href="#">4.</a>	<a href="#">Atom Element Definitions . . . . .</a>	<a href="#">13</a>
<a href="#">4.1</a>	<a href="#">Container Elements . . . . .</a>	<a href="#">13</a>
<a href="#">4.1.1</a>	<a href="#">The "atom:feed" Element . . . . .</a>	<a href="#">13</a>
<a href="#">4.1.2</a>	<a href="#">The "atom:entry" Element . . . . .</a>	<a href="#">15</a>
<a href="#">4.1.3</a>	<a href="#">The "atom:content" Element . . . . .</a>	<a href="#">17</a>
<a href="#">4.2</a>	<a href="#">Metadata Elements . . . . .</a>	<a href="#">20</a>
<a href="#">4.2.1</a>	<a href="#">The "atom:author" Element . . . . .</a>	<a href="#">20</a>
<a href="#">4.2.2</a>	<a href="#">The "atom:category" Element . . . . .</a>	<a href="#">20</a>
<a href="#">4.2.3</a>	<a href="#">The "atom:contributor" Element . . . . .</a>	<a href="#">21</a>
<a href="#">4.2.4</a>	<a href="#">The "atom:generator" Element . . . . .</a>	<a href="#">21</a>
<a href="#">4.2.5</a>	<a href="#">The "atom:icon" Element . . . . .</a>	<a href="#">21</a>
<a href="#">4.2.6</a>	<a href="#">The "atom:id" Element . . . . .</a>	<a href="#">22</a>
<a href="#">4.2.7</a>	<a href="#">The "atom:link" Element . . . . .</a>	<a href="#">24</a>
<a href="#">4.2.8</a>	<a href="#">The "atom:logo" Element . . . . .</a>	<a href="#">26</a>
<a href="#">4.2.9</a>	<a href="#">The "atom:published" Element . . . . .</a>	<a href="#">26</a>
<a href="#">4.2.10</a>	<a href="#">The "atom:rights" Element . . . . .</a>	<a href="#">26</a>
<a href="#">4.2.11</a>	<a href="#">The "atom:source" Element . . . . .</a>	<a href="#">27</a>
<a href="#">4.2.12</a>	<a href="#">The "atom:subtitle" Element . . . . .</a>	<a href="#">27</a>
<a href="#">4.2.13</a>	<a href="#">The "atom:summary" Element . . . . .</a>	<a href="#">28</a>
<a href="#">4.2.14</a>	<a href="#">The "atom:title" Element . . . . .</a>	<a href="#">28</a>
<a href="#">4.2.15</a>	<a href="#">The "atom:updated" Element . . . . .</a>	<a href="#">28</a>
<a href="#">5.</a>	<a href="#">Securing Atom Documents . . . . .</a>	<a href="#">29</a>
<a href="#">6.</a>	<a href="#">Extending Atom . . . . .</a>	<a href="#">32</a>
<a href="#">7.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">34</a>
<a href="#">8.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">36</a>
<a href="#">9.</a>	<a href="#">References . . . . .</a>	<a href="#">38</a>
<a href="#">9.1</a>	<a href="#">Normative References . . . . .</a>	<a href="#">38</a>
<a href="#">9.2</a>	<a href="#">Informative References . . . . .</a>	<a href="#">39</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">40</a>
<a href="#">A.</a>	<a href="#">Contributors . . . . .</a>	<a href="#">41</a>
<a href="#">B.</a>	<a href="#">RELAX NG Compact Schema . . . . .</a>	<a href="#">42</a>
<a href="#">C.</a>	<a href="#">Change Log . . . . .</a>	<a href="#">50</a>
	<a href="#">Intellectual Property and Copyright Statements . . . . .</a>	<a href="#">56</a>



## **1. Introduction**

Atom is an XML-based document format that describes lists of related information known as "feeds". Feeds are composed of a number of items, known as "entries", each with an extensible set of attached metadata. For example, each entry has a title.

The primary use case that Atom addresses is the syndication of Web content such as Weblogs and news headlines to Web sites as well as directly to user agents.

### **1.1 Examples**

A brief, single-entry Atom Feed Document:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Example Feed</title>
  <link href="http://example.org/" />
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03" />
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>

</feed>
```



A more extensive, single-entry Atom Feed Document:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">dive into mark</title>
  <subtitle type="html">
    A &lt;em>lot</em> of effort
    went into making this effortless
  </subtitle>
  <updated>2005-07-31T12:29:29Z</updated>
  <id>tag:example.org,2003:3</id>
  <link rel="alternate" type="text/html"
    hreflang="en" href="http://example.org/" />
  <link rel="self" type="application/atom+xml"
    href="http://example.org/feed.atom" />
  <rights>Copyright (c) 2003, Mark Pilgrim</rights>
  <generator uri="http://www.example.com/" version="1.0">
    Example Toolkit
  </generator>
  <entry>
    <title>Atom draft-07 snapshot</title>
    <link rel="alternate" type="text/html"
      href="http://example.org/2005/04/02/atom" />
    <link rel="enclosure" type="audio/mpeg" length="1337"
      href="http://example.org/audio/ph34r_my_podcast.mp3" />
    <id>tag:example.org,2003:3.2397</id>
    <updated>2005-07-31T12:29:29Z</updated>
    <published>2003-12-13T08:29:29-04:00</published>
    <author>
      <name>Mark Pilgrim</name>
      <uri>http://example.org/</uri>
      <email>f8dy@example.com</email>
    </author>
    <contributor>
      <name>Sam Ruby</name>
    </contributor>
    <contributor>
      <name>Joe Gregorio</name>
    </contributor>
    <content type="xhtml" xml:lang="en"
      xml:base="http://diveintomark.org/">
      <div xmlns="http://www.w3.org/1999/xhtml">
        <p><i>[Update: The Atom draft is finished.]</i></p>
      </div>
    </content>
  </entry>
</feed>
```



## **1.2 Namespace and Version**

The XML Namespaces URI [[W3C.REC-xml-names-19990114](#)] for the XML data format described in this specification is:

<http://www.w3.org/2005/Atom>

For convenience, this data format may be referred to as "Atom 1.0". This specification uses "Atom" internally.

## **1.3 Notational Conventions**

This specification describes conformance in terms of two artifacts; Atom Feed Documents and Atom Entry Documents. Additionally, it places some requirements on Atom Processors.

This specification uses the namespace prefix "atom:" for the Namespace URI identified in [section 1.2](#). above. Note that the choice of namespace prefix is arbitrary and not semantically significant.

Atom is specified using terms from the XML Infoset [[W3C.REC-xml-infoset-20040204](#)]. However, this specification uses a shorthand for two common terms; the phrase "Information Item" is omitted when naming Element Information Items and Attribute Information Items. Therefore, when this specification uses the term "element," it is referring to an Element Information Item in Infoset terms. Likewise, when it uses the term "attribute," it is referring to an Attribute Information Item.

Some sections of this specification are illustrated with fragments of a non-normative RELAX NG Compact schema [[RELAX-NG](#)]. However, the text of this specification provides the definition of conformance. A complete schema appears in [Appendix B](#).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#), [[RFC2119](#)], as scoped to those conformance targets.





## 2. Atom Documents

This specification describes two kinds of Atom Documents; Atom Feed Documents and Atom Entry Documents.

An Atom Feed Document is a representation of an Atom feed, including metadata about the feed, and some or all of the entries associated with it. Its root is the atom:feed element.

An Atom Entry Document represents exactly one Atom entry, outside of the context of an Atom feed. Its root is the atom:entry element.

```
namespace atom = "http://www.w3.org/2005/Atom"
start = atomFeed | atomEntry
```

Both kinds of Atom Documents are specified in terms of the XML Information Set, serialised as XML 1.0 [[W3C.REC-xml-20040204](#)] and identified with the "application/atom+xml" media type. Atom Documents MUST be well-formed XML. This specification does not define a DTD for Atom Documents, and hence does not require them to be valid (in the sense used by XML).

Atom allows the use of IRIs [[RFC3987](#)]. Every URI [[RFC3986](#)] is also an IRI, so a URI may be used wherever below an IRI is named. There are two special considerations: when an IRI which is not also a URI is given for dereferencing, it MUST be mapped to a URI using the steps in [Section 3.1 of \[RFC3987\]](#); when an IRI is serving as an atom:id value, it MUST NOT be so mapped, so that the comparison works as described in [Section 4.2.6.1](#).

Any element defined by this specification MAY have an xml:base attribute [[W3C.REC-xmlbase-20010627](#)]. When xml:base is used in an Atom Document, it serves the function described in [section 5.1.1 of \[RFC3986\]](#), establishing the base URI (or IRI) for resolving any relative references found within the effective scope of the xml:base attribute.

Any element defined by this specification MAY have an xml:lang attribute, whose content indicates the natural language for the element and its descendents. The language context is only significant for elements and attributes declared to be "Language-Sensitive" by this specification. Requirements regarding the content and interpretation of xml:lang are specified in XML 1.0 [[W3C.REC-xml-20040204](#)], [Section 2.12](#).

```
atomCommonAttributes =
  attribute xml:base { atomUri }?,
  attribute xml:lang { atomLanguageTag }?,
```



undefinedAttribute\*

Atom is an extensible format. See [Section 6](#) of this document for a full description of how Atom Documents can be extended.

Atom Processors MAY keep state sourced from Atom Feed Documents and combine them with other Atom Feed Documents, in order to facilitate a contiguous view of the contents of a feed. The manner in which Atom Feed Documents are combined in order to reconstruct a feed (e.g., updating entries and metadata, dealing with missing entries) is out of the scope of this specification.

### **3. Common Atom Constructs**

Many of Atom's elements share a few common structures. This section defines those structures and their requirements for convenient reference by the appropriate element definitions.

When an element is identified as being a particular kind of construct, it inherits the corresponding requirements from that construct's definition in this section.

Note that there **MUST NOT** be any whitespace in a Date construct or in any IRI. Some XML-emitting implementations erroneously insert whitespace around values by default, and such implementations will emit invalid Atom Documents.

#### **3.1 Text Constructs**

A Text construct contains human-readable text, usually in small quantities. The content of Text constructs is Language-Sensitive.

```
atomPlainTextConstruct =  
  atomCommonAttributes,  
  attribute type { "text" | "html" }?,  
  text
```

```
atomXHTMLTextConstruct =  
  atomCommonAttributes,  
  attribute type { "xhtml" },  
  xhtmlDiv
```

```
atomTextConstruct = atomPlainTextConstruct | atomXHTMLTextConstruct
```

##### **3.1.1 The "type" Attribute**

Text constructs **MAY** have a "type" attribute. When present, the value **MUST** be one of "text", "html" or "xhtml". If the "type" attribute is not provided, Atom Processors **MUST** behave as though it were present with a value of "text". Unlike the atom:content element defined in [Section 4.1.3](#), MIME media types [[MIMEREG](#)] **MUST NOT** be used as values for the "type" attribute on Text constructs.



#### [3.1.1.1](#) Text

Example atom:title with text content:

```
...  
<title type="text">  
  Less: &lt;  
</title>  
...
```

If the value is "text", the content of the Text construct MUST NOT contain child elements. Such text is intended to be presented to humans in a readable fashion. Thus, Atom Processors MAY collapse white-space (including line-breaks), and display the text using typographic techniques such as justification and proportional fonts.

#### [3.1.1.2](#) HTML

Example atom:title with HTML content:

```
...  
<title type="html">  
  Less: &lt;em> &amp;lt; &lt;/em>  
</title>  
...
```

If the value of "type" is "html", the content of the Text construct MUST NOT contain child elements, and SHOULD be suitable for handling as HTML [[HTML](#)]. Any markup within MUST be escaped; for example, "<br>" as "&lt;br>". HTML markup within SHOULD be such that it could validly appear directly within an HTML <DIV> element, after unescaping. Atom Processors that display such content MAY use that markup to aid in its display.





### [3.1.1.3](#) XHTML

Example atom:title with XHTML content:

```
...
<title type="xhtml" xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <xhtml:div>
    Less: <xhtml:em> &lt; </xhtml:em>
  </xhtml:div>
</title>
...
```

If the value of "type" is "xhtml", the content of the Text construct MUST be a single XHTML div element [[XHTML](#)], and SHOULD be suitable for handling as XHTML. The XHTML div element itself MUST NOT be considered part of the content. Atom Processors that display the content MAY use the markup to aid in displaying it. The escaped versions of characters such as "&" and ">" represent those characters, not markup.

Examples of valid XHTML content:

```
...
<summary type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
    This is <b>XHTML</b> content.
  </div>
</summary>
...
<summary type="xhtml">
  <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">
    This is <xhtml:b>XHTML</xhtml:b> content.
  </xhtml:div>
</summary>
...
```

The following example assumes that the XHTML namespace has been bound to the "xh" prefix earlier in the document:

```
...
<summary type="xhtml">
  <xh:div>
    This is <xh:b>XHTML</xh:b> content.
  </xh:div>
</summary>
...
```



### **[3.2](#) Person Constructs**

A Person construct is an element that describes a person, corporation, or similar entity (hereafter, 'person').

```
atomPersonConstruct =  
  atomCommonAttributes,  
  (element atom:name { text }  
    & element atom:uri { atomUri }?  
    & element atom:email { atomEmailAddress }?  
    & extensionElement*)
```

This specification assigns no significance to the order of appearance of the child elements in a Person construct. Person constructs allow extension Metadata elements (see [Section 6.4](#)).

#### **[3.2.1](#) The "atom:name" Element**

The "atom:name" element's content conveys a human-readable name for the person. The content of atom:name is Language-Sensitive. Person constructs MUST contain exactly one "atom:name" element.

#### **[3.2.2](#) The "atom:uri" Element**

The "atom:uri" element's content conveys an IRI associated with the person. Person constructs MAY contain an atom:uri element, but MUST NOT contain more than one. The content of atom:uri in a Person construct MUST be an IRI reference [[RFC3987](#)].

#### **[3.2.3](#) The "atom:email" Element**

The "atom:email" element's content conveys an e-mail address associated with the person. Person constructs MAY contain an atom:email element, but MUST NOT contain more than one. Its content MUST conform to the "addr-spec" production in [[RFC2822](#)].



### 3.3 Date Constructs

A Date construct is an element whose content MUST conform to the "date-time" production in [[RFC3339](#)]. In addition, an uppercase "T" character MUST be used to separate date and time, and an uppercase "Z" character MUST be present in the absence of a numeric time zone offset.

```
atomDateConstruct =  
    atomCommonAttributes,  
    xsd:dateTime
```

Such date values happen to be compatible with the following specifications: [[ISO.8601.1988](#)], [[W3C.NOTE-datetime-19980827](#)], and [[W3C.REC-xmlschema-2-20041028](#)].

Example Date constructs:

```
<updated>2003-12-13T18:30:02Z</updated>  
<updated>2003-12-13T18:30:02.25Z</updated>  
<updated>2003-12-13T18:30:02+01:00</updated>  
<updated>2003-12-13T18:30:02.25+01:00</updated>
```

Date values SHOULD be as accurate as possible. For example, it would be generally inappropriate for a publishing system to apply the same timestamp to several entries which were published during the course of a single day.



## [4.](#) Atom Element Definitions

### [4.1](#) Container Elements

#### [4.1.1](#) The "atom:feed" Element

The "atom:feed" element is the document (i.e., top-level) element of an Atom Feed Document, acting as a container for metadata and data associated with the feed. Its element children consist of metadata elements followed by zero or more atom:entry child elements.

```
atomFeed =  
  element atom:feed {  
    atomCommonAttributes,  
    (atomAuthor*  
      & atomCategory*  
      & atomContributor*  
      & atomGenerator?  
      & atomIcon?  
      & atomId  
      & atomLink*  
      & atomLogo?  
      & atomRights?  
      & atomSubtitle?  
      & atomTitle  
      & atomUpdated  
      & extensionElement*),  
    atomEntry*  
  }
```

This specification assigns no significance to the order of atom:entry elements within the feed.

The following child elements are defined by this specification (note that the presence of some of these elements is required):

- o atom:feed elements MUST contain one or more atom:author elements, unless all of the atom:feed element's child atom:entry elements contain at least one atom:author element.
- o atom:feed elements MAY contain any number of atom:category elements.
- o atom:feed elements MAY contain any number of atom:contributor elements.
- o atom:feed elements MUST NOT contain more than one atom:generator element.





- o atom:feed elements MUST NOT contain more than one atom:icon element.
- o atom:feed elements MUST NOT contain more than one atom:logo element.
- o atom:feed elements MUST contain exactly one atom:id element.
- o atom:feed elements SHOULD contain one atom:link element with a rel attribute value of "self". This is the preferred URI for retrieving Atom Feed Documents representing this Atom feed.
- o atom:feed elements MUST NOT contain more than one atom:link element with a rel attribute value of "alternate" that has the same combination of type and hreflang attribute values.
- o atom:feed elements MAY contain additional atom:link elements beyond those described above.
- o atom:feed elements MUST NOT contain more than one atom:rights element.
- o atom:feed elements MUST NOT contain more than one atom:subtitle element.
- o atom:feed elements MUST contain exactly one atom:title element.
- o atom:feed elements MUST contain exactly one atom:updated element.

If multiple atom:entry elements with the same atom:id value appear in an Atom Feed Document, they represent the same entry. Their atom:updated timestamps SHOULD be different. If an Atom Feed Document contains multiple entries with the same atom:id, Atom Processors MAY choose to display all of them or some subset of them. One typical behavior would be to display only the entry with the latest atom:updated timestamp.

#### **4.1.1.1 Providing Textual Content**

Experience teaches that feeds which contain textual content are in general more useful than those which do not. Some applications (one example is full-text indexers) require a minimum amount of text or (X)HTML to function reliably and predictably. Feed producers should be aware of these issues. It is advisable that each atom:entry element contain a non-empty atom:title element, a non-empty atom:content element when that element is present, and a non-empty atom:summary element when the entry contains no atom:content element. However, the absence of atom:summary is not an error and Atom



Processors MUST NOT fail to function correctly as a consequence of such an absence.

#### [4.1.2](#) The "atom:entry" Element

The "atom:entry" element represents an individual entry, acting as a container for metadata and data associated with the entry. This element can appear as a child of the atom:feed element, or it can appear as the document (i.e., top-level) element of a standalone Atom Entry Document.

```
atomEntry =
  element atom:entry {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContent?
     & atomContributor*
     & atomId
     & atomLink*
     & atomPublished?
     & atomRights?
     & atomSource?
     & atomSummary?
     & atomTitle
     & atomUpdated
     & extensionElement*)
  }
```

This specification assigns no significance to the order of appearance of the child elements of atom:entry.

The following child elements are defined by this specification (note that it requires the presence of some of these elements):

- o atom:entry elements MUST contain one or more atom:author elements, unless the atom:entry contains an atom:source element which contains an atom:author element, or, in an Atom Feed Document, the atom:feed element contains an atom:author element itself.
- o atom:entry elements MAY contain any number of atom:category elements.
- o atom:entry elements MUST NOT contain more than one atom:content element.
- o atom:entry elements MAY contain any number of atom:contributor elements.



- o atom:entry elements MUST contain exactly one atom:id element.
- o atom:entry elements that contain no child atom:content element MUST contain at least one atom:link element with a rel attribute value of "alternate".
- o atom:entry elements MUST NOT contain more than one atom:link element with a rel attribute value of "alternate" that has the same combination of type and hreflang attribute values.
- o atom:entry elements MAY contain additional atom:link elements beyond those described above.
- o atom:entry elements MUST NOT contain more than one atom:published element.
- o atom:entry elements MUST NOT contain more than one atom:rights element.
- o atom:entry elements MUST NOT contain more than one atom:source element.
- o atom:entry elements MUST contain an atom:summary element in either of the following cases:
  - \* the atom:entry contains an atom:content that has a "src" attribute (and is thus empty).
  - \* the atom:entry contains content that is encoded in Base64; i.e. the "type" attribute of atom:content is a MIME media type [[MIMEREG](#)], but is not an XML media type [[RFC3023](#)], does not begin with "text/", and does not end with "/xml" or "+xml".
- o atom:entry elements MUST NOT contain more than one atom:summary element.
- o atom:entry elements MUST contain exactly one atom:title element.
- o atom:entry elements MUST contain exactly one atom:updated element.



#### [4.1.3](#) The "atom:content" Element

The "atom:content" element either contains or links to the content of the entry. The content of atom:content is Language-Sensitive.

```
atomInlineTextContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { "text" | "html" }?,
    (text)*
  }

atomInlineXHTMLContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { "xhtml" },
    xhtmlDiv
  }

atomInlineOtherContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    (text|anyElement)*
  }

atomOutOfLineContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    attribute src { atomUri },
    empty
  }

atomContent = atomInlineTextContent
| atomInlineXHTMLContent
| atomInlineOtherContent
| atomOutOfLineContent
```

##### [4.1.3.1](#) The "type" attribute

On the atom:content element, the value of the "type" attribute MAY be one of "text", "html", or "xhtml". Failing that, it MUST conform to the syntax of a MIME media type, but MUST NOT be a composite type (see Section 4.2.6 of [[MIMEREG](#)]). If neither the type attribute nor the src attribute is provided, Atom Processors MUST behave as though the type attribute were present with a value of "text".





#### **4.1.3.2 The "src" attribute**

atom:content MAY have a "src" attribute, whose value MUST be an IRI reference [[RFC3987](#)]. If the "src" attribute is present, atom:content MUST be empty. Atom Processors MAY use the IRI to retrieve the content, and MAY chose to ignore remote content or present it in a different manner than local content.

If the "src" attribute is present, the "type" attribute SHOULD be provided and MUST be a MIME media type [[MIMEREG](#)], rather than "text", "html", or "xhtml". The value is advisory; that is to say, when the corresponding URI (mapped from an IRI, if necessary), is dereferenced, if the server providing that content also provides a media type, the server-provided media type is authoritative.

#### **4.1.3.3 Processing Model**

Atom Documents MUST conform to the following rules. Atom Processors MUST interpret atom:content according to the first applicable rule.

1. If the value of "type" is "text", the content of atom:content MUST NOT contain child elements. Such text is intended to be presented to humans in a readable fashion. Thus, Atom Processors MAY collapse white-space (including line-breaks), and display the text using typographic techniques such as justification and proportional fonts.
2. If the value of "type" is "html", the content of atom:content MUST NOT contain child elements, and SHOULD be suitable for handling as HTML [[HTML](#)]. The HTML markup MUST be escaped; for example, "<br>" as "&lt;br>". The HTML markup SHOULD be such that it could validly appear directly within an HTML <DIV> element. Atom Processors that display the content MAY use the markup to aid in displaying it.
3. If the value of "type" is "xhtml", the content of atom:content MUST be a single XHTML div element [[XHTML](#)], and SHOULD be suitable for handling as XHTML. The XHTML div element itself MUST NOT be considered part of the content. Atom Processors that display the content MAY use the markup to aid in displaying it. The escaped versions of characters such as "&" and ">" represent those characters, not markup.
4. If the value of "type" is an XML media type [[RFC3023](#)], or ends with "+xml" or "/xml" (case-insensitive), the content of atom:content MAY include child elements, and SHOULD be suitable for handling as the indicated media type. If the "src" attribute is not provided, this would normally mean that the "atom:content"



element would contain a single child element which would serve as the root element of the XML document of the indicated type.

5. If the value of "type" begins with "text/" (case-insensitive), the content of atom:content MUST NOT contain child elements.
6. For all other values of "type", the content of atom:content MUST be a valid Base64 encoding, as described in [\[RFC3548\]](#), [section 3](#). When decoded, it SHOULD be suitable for handling as the indicated media type. In this case, the characters in the Base64 encoding MAY be preceded and followed in the atom:content element by white-space, and lines are separated by a single newline (U+000A) character.

#### [4.1.3.4](#) Examples

XHTML inline:

```
...
<content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
    This is <b>XHTML</b> content.
  </div>
</content>
...
<content type="xhtml">
  <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">
    This is <xhtml:b>XHTML</xhtml:b> content.
  </xhtml:div>
</content>
...
```

The following example assumes that the XHTML namespace has been bound to the "xh" prefix earlier in the document:

```
...
<content type="xhtml">
  <xh:div>
    This is <xh:b>XHTML</xh:b> content.
  </xh:div>
</content>
...
```



## [4.2](#) Metadata Elements

### [4.2.1](#) The "atom:author" Element

The "atom:author" element is a Person construct that indicates the author of the entry or feed.

```
atomAuthor = element atom:author { atomPersonConstruct }
```

If an atom:entry element does not contain atom:author elements, then the atom:author elements of the contained atom:source element are considered to apply. In an Atom Feed Document, the atom:author elements of the containing atom:feed element are considered to apply to the entry if there are no atom:author elements in the locations described above.

### [4.2.2](#) The "atom:category" Element

The "atom:category" element conveys information about a category associated with an entry or feed. This specification assigns no meaning to the content (if any) of this element.

```
atomCategory =  
  element atom:category {  
    atomCommonAttributes,  
    attribute term { text },  
    attribute scheme { atomUri }?,  
    attribute label { text }?,  
    undefinedContent  
  }
```

#### [4.2.2.1](#) The "term" Attribute

The "term" attribute is a string that identifies the category to which the entry or feed belongs. Category elements MUST have a "term" attribute.

#### [4.2.2.2](#) The "scheme" Attribute

The "scheme" attribute is an IRI that identifies a categorization scheme. Category elements MAY have a "scheme" attribute.

#### [4.2.2.3](#) The "label" attribute

The "label" attribute provides a human-readable label for display in end-user applications. The content of the "label" attribute is Language-Sensitive. Entities such as "&#" and "&lt;" represent their corresponding characters ("&" and "<" respectively), not



markup. Category elements MAY have a "label" attribute.

#### [4.2.3](#) The "atom:contributor" Element

The "atom:contributor" element is a Person construct that indicates a person or other entity who contributed to the entry or feed.

```
atomContributor = element atom:contributor { atomPersonConstruct }
```

#### [4.2.4](#) The "atom:generator" Element

The "atom:generator" element's content identifies the agent used to generate a feed, for debugging and other purposes.

```
atomGenerator = element atom:generator {  
    atomCommonAttributes,  
    attribute uri { atomUri }?,  
    attribute version { text }?,  
    text  
}
```

The content of this element, when present, MUST be a string that is a human-readable name for the generating agent. Entities such as "&" and "<" represent their corresponding characters ("&" and "<" respectively), not markup.

The atom:generator element MAY have a "uri" attribute whose value MUST be an IRI reference [[RFC3987](#)]. When dereferenced, the resulting URI (mapped from an IRI, if necessary) SHOULD produce a representation that is relevant to that agent.

The atom:generator element MAY have a "version" attribute that indicates the version of the generating agent.

#### [4.2.5](#) The "atom:icon" Element

The "atom:icon" element's content is an IRI reference [[RFC3987](#)] which identifies an image which provides iconic visual identification for a feed.

```
atomIcon = element atom:icon {  
    atomCommonAttributes,  
    (atomUri)  
}
```

The image SHOULD have an aspect ratio of one (horizontal) to one (vertical), and SHOULD be suitable for presentation at a small size.





#### **4.2.6 The "atom:id" Element**

The "atom:id" element conveys a permanent, universally unique identifier for an entry or feed.

```
atomId = element atom:id {  
    atomCommonAttributes,  
    (atomUri)  
}
```

Its content MUST be an IRI, as defined by [[RFC3987](#)]. Note that the definition of "IRI" excludes relative references. Though the IRI might use a dereferencable scheme, Atom Processors MUST NOT assume it can be dereferenced.

When an Atom Document is relocated, migrated, syndicated, republished, exported or imported, the content of its atom:id element MUST NOT change. Put another way, an atom:id element pertains to all instantiations of a particular Atom entry or feed; revisions retain the same content in their atom:id elements. It is suggested that the atom:id element be stored along with the associated resource.

The content of an atom:id element MUST be created in a way that assures uniqueness.

Because of the risk of confusion between IRIs that would be equivalent if mapped to URIs and dereferenced, the following normalization strategy SHOULD be applied when generating atom:id elements:

- o Provide the scheme in lowercase characters.
- o Provide the host, if any, in lowercase characters.
- o Only perform percent-encoding where it is essential.
- o Use uppercase A-through-F characters when percent-encoding.
- o Prevent dot-segments appearing in paths.
- o For schemes that define a default authority, use an empty authority if the default is desired.
- o For schemes that define an empty path to be equivalent to a path of "/", use "/".
- o For schemes that define a port, use an empty port if the default is desired.



- o Preserve empty fragment identifiers and queries.
- o Ensure that all components of the IRI are appropriately character-normalized, e.g. by using NFC or NFKC.

#### **4.2.6.1 Comparing atom:id**

Instances of atom:id elements can be compared to determine whether an entry or feed is the same as one seen before. Processors MUST compare atom:id elements on a character-by-character basis (in a case-sensitive fashion). Comparison operations MUST be based solely on the IRI character strings, and MUST NOT rely on dereferencing the IRIs or URIs mapped from them.

As a result, two IRIs that resolve to the same resource but are not character-for-character identical will be considered different for the purposes of identifier comparison.

For example, these are four distinct identifiers, despite the fact that they differ only in case:

`http://www.example.org/thing`

`http://www.example.org/Thing`

`http://www.EXAMPLE.org/thing`

`HTTP://www.example.org/thing`

Likewise, these are three distinct identifiers, because IRI %-escaping is significant for the purposes of comparison:

`http://www.example.com/~bob`

`http://www.example.com/%7ebob`

`http://www.example.com/%7Ebob`



#### [4.2.7](#) The "atom:link" Element

The "atom:link" element defines a reference from an entry or feed to a Web resource. This specification assigns no meaning to the content (if any) of this element.

```
atomLink =  
  element atom:link {  
    atomCommonAttributes,  
    attribute href { atomUri },  
    attribute rel { atomNCName | atomUri }?,  
    attribute type { atomMediaType }?,  
    attribute hreflang { atomLanguageTag }?,  
    attribute title { text }?,  
    attribute length { text }?,  
    undefinedContent  
  }
```

##### [4.2.7.1](#) The "href" Attribute

The "href" attribute contains the link's IRI. atom:link elements MUST have a href attribute, whose value MUST be a IRI reference [[RFC3987](#)].

##### [4.2.7.2](#) The "rel" Attribute

atom:link elements MAY have a "rel" attribute that indicates the link relation type. If the "rel" attribute is not present, the link element MUST be interpreted as if the link relation type is "alternate".

The value of "rel" MUST be a string that is non-empty, and matches either the "isegment-nz-nc" or the "IRI" production in [[RFC3987](#)]. Note that use of a relative reference other than a simple name is not allowed. If a name is given, implementations MUST consider the link relation type to be equivalent to the same name registered within the IANA Registry of Link Relations ([Section 7](#)), and thus the IRI that would be obtained by appending the value of the rel attribute to the string "http://www.iana.org/assignments/relation/". The value of "rel" describes the meaning of the link, but does not impose any behavioral requirements on Atom Processors.

This document defines five initial values for the Registry of Link Relations:

1. The value "alternate" signifies that the IRI in the value of the href attribute identifies an alternate version of the resource described by the containing element.



2. The value "related" signifies that the IRI in the value of the href attribute identifies a resource related to the resource described by the containing element. For example, the feed for a site that discusses the performance of the search engine at "http://search.example.com" might contain, as a child of atom:feed:

```
<link rel="related" href="http://search.example.com/" />
```

An identical link might appear as a child of any atom:entry whose content contains a discussion of that same search engine.

3. The value "self" signifies that the IRI in the value of the href attribute identifies a resource equivalent to the containing element.
4. The value "enclosure" signifies that the IRI in the value of the href attribute identifies a related resource which is potentially large in size and might require special handling. For atom:link elements with rel="enclosure", the length attribute SHOULD be provided.
5. The value "via" signifies that the IRI in the value of the href attribute identifies a resource that is the source of the information provided in the containing element.

#### **4.2.7.3 The "type" Attribute**

On the link element, the "type" attribute's value is an advisory media type; it is a hint about the type of the representation that is expected to be returned when the value of the href attribute is dereferenced. Note that the type attribute does not override the actual media type returned with the representation. Link elements MAY have a type attribute, whose value MUST conform to the syntax of a MIME media type [[MIMEREG](#)].

#### **4.2.7.4 The "hreflang" Attribute**

The "hreflang" attribute's content describes the language of the resource pointed to by the href attribute. When used together with the rel="alternate", it implies a translated version of the entry. Link elements MAY have an hreflang attribute, whose value MUST be a language tag [[RFC3066](#)].

#### **4.2.7.5 The "title" Attribute**

The "title" attribute conveys human-readable information about the





link. The content of the "title" attribute is Language-Sensitive. Entities such as "&" and "<" represent their corresponding characters ("&" and "<" respectively), not markup. Link elements MAY have a title attribute.

#### [4.2.7.6](#) The "length" Attribute

The "length" attribute indicates an advisory length of the linked content in octets; it is a hint about the content length of the representation returned when the IRI in the href attribute is mapped to a URI and dereferenced. Note that the length attribute does not override the actual content length of the representation as reported by the underlying protocol. Link elements MAY have a length attribute.

#### [4.2.8](#) The "atom:logo" Element

The "atom:logo" element's content is an IRI reference [[RFC3987](#)] which identifies an image which provides visual identification for a feed.

```
atomLogo = element atom:logo {  
    atomCommonAttributes,  
    (atomUri)  
}
```

The image SHOULD have an aspect ratio of 2 (horizontal) to 1 (vertical).

#### [4.2.9](#) The "atom:published" Element

The "atom:published" element is a Date construct indicating an instant in time associated with an event early in the life cycle of the entry.

```
atomPublished = element atom:published { atomDateConstruct }
```

Typically, atom:published will be associated with the initial creation or first availability of the resource.

#### [4.2.10](#) The "atom:rights" Element

The "atom:rights" element is a Text construct that conveys information about rights held in and over an entry or feed.

```
atomRights = element atom:rights { atomTextConstruct }
```

The atom:rights element SHOULD NOT be used to convey machine-readable licensing information.



If an atom:entry element does not contain an atom:rights element, then the atom:rights element of the containing atom:feed element, if present, is considered to apply to the entry.

#### [4.2.11](#) The "atom:source" Element

If an atom:entry is copied from one feed into another feed, then the source atom:feed's metadata (all child elements of atom:feed other than the atom:entry elements) MAY be preserved within the copied entry by adding an atom:source child element, if it is not already present in the entry, and including some or all of the source feed's Metadata elements as the atom:source element's children. Such metadata SHOULD be preserved if the source atom:feed contains any of the child elements atom:author, atom:contributor, atom:rights, or atom:category and those child elements are not present in the source atom:entry.

```
atomSource =  
  element atom:source {  
    atomCommonAttributes,  
    (atomAuthor*  
      & atomCategory*  
      & atomContributor*  
      & atomGenerator?  
      & atomIcon?  
      & atomId?  
      & atomLink*  
      & atomLogo?  
      & atomRights?  
      & atomSubtitle?  
      & atomTitle?  
      & atomUpdated?  
      & extensionElement*)  
  }
```

The atom:source element is designed to allow the aggregation of entries from different feeds while retaining information about an entry's source feed. For this reason, Atom Processors which are performing such aggregation SHOULD include at least the required feed-level Metadata elements (atom:id, atom:title, and atom:updated) in the atom:source element.

#### [4.2.12](#) The "atom:subtitle" Element

The "atom:subtitle" element is a Text construct that conveys a human-readable description or subtitle for a feed.

```
atomSubtitle = element atom:subtitle { atomTextConstruct }
```



#### [4.2.13](#) The "atom:summary" Element

The "atom:summary" element is a Text construct that conveys a short summary, abstract, or excerpt of an entry.

```
atomSummary = element atom:summary { atomTextConstruct }
```

It is not advisable for the atom:summary element to duplicate atom:title or atom:content, because Atom Processors might assume there is a useful summary when there is none.

#### [4.2.14](#) The "atom:title" Element

The "atom:title" element is a Text construct that conveys a human-readable title for an entry or feed.

```
atomTitle = element atom:title { atomTextConstruct }
```

#### [4.2.15](#) The "atom:updated" Element

The "atom:updated" element is a Date construct indicating the most recent instant in time when an entry or feed was modified in a way the publisher considers significant. Therefore, not all modifications necessarily result in a changed atom:updated value.

```
atomUpdated = element atom:updated { atomDateConstruct }
```

Publishers MAY change the value of this element over time.



## **5. Securing Atom Documents**

Because Atom is an XML-based format, existing XML security mechanisms can be used to secure its content.

Producers of feeds and/or entries, and intermediaries who aggregate feeds and/or entries, may have sound reasons for signing and/or encrypting otherwise-unprotected content. For example, a merchant might digitally sign a message that contains a discount coupon for its products. A bank that uses Atom to deliver customer statements is very likely to want to sign and encrypt those messages to protect their customers' financial information and to assure the customer of their authenticity. Intermediaries may want to encrypt aggregated feeds so that a passive observer cannot tell what topics the recipient is interested in. Of course, many other examples exist as well.

The algorithm requirements in this section pertain to the Atom Processor. They require that a recipient, at a minimum, be able to handle messages that use the specified cryptographic algorithms. These requirements do not limit the algorithms that the sender can choose.

### **5.1 Digital Signatures**

The root of an Atom Document (i.e., atom:feed in an Atom Feed Document, atom:entry in an Atom Entry Document), or any atom:entry element, MAY have an Enveloped Signature, as described by XML-Signature and Syntax Processing [[W3C.REC-xmldsig-core-20020212](#)].

Atom Processors MUST NOT reject an Atom Document containing such a signature because they are not capable of verifying it; they MUST continue processing and MAY inform the user of their failure to validate the signature.

In other words, the presence of an element with the namespace URI "http://www.w3.org/2000/09/xmldsig#" and a local name of "Signature" as a child of the document element MUST NOT cause an Atom Processor to fail merely because of its presence.

Other elements in an Atom Document MUST NOT be signed unless their definitions explicitly specify such a capability.

Section 6.5.1 of [[W3C.REC-xmldsig-core-20020212](#)] requires support for Canonical XML [[W3C.REC-xml-c14n-20010315](#)]. However, many implementers do not use it because signed XML documents enclosed in other XML documents have their signatures broken. Thus, Atom Processors that verify signed Atom Documents MUST be able to





canonicalize with the exclusive XML canonicalization method identified by the URI "<http://www.w3.org/2001/10/xml-exc-c14n#>", as specified in Exclusive XML Canonicalization [W3C.REC-xml-exc-c14n-20020718].

Intermediaries such as aggregators may need to add an atom:source element to an entry that does not contain its own atom:source element. If such an entry is signed, the addition will break the signature. Thus, a publisher of individually-signed entries should strongly consider adding an atom:source element to those entries before signing them. Implementors should also be aware of the issues concerning the use of markup in the "xml:" namespace as it interacts with canonicalization.

Section 4.4.2 of [[W3C.REC-xmlsig-core-20020212](#)] requires support for DSA signatures and recommends support for RSA signatures. However, because of the much greater popularity in the market of RSA versus DSA, Atom Processors that verify signed Atom Documents MUST be able to verify RSA signatures, but do not need be able to verify DSA signatures. Due to security issues that can arise if the keying material for MAC (message authentication code) authentication is not handled properly, Atom Documents SHOULD NOT use MACs for signatures.

## 5.2 Encryption

The root of an Atom Document (i.e., atom:feed in an Atom Feed Document, atom:entry in an Atom Entry Document) MAY be encrypted, using the mechanisms described by XML Encryption Syntax and Processing [[W3C.REC-xmlenc-core-20021210](#)].

Section 5.1 of [[W3C.REC-xmlenc-core-20021210](#)] requires support of TripleDES, AES-128, and AES-256. Atom Processors that decrypt Atom Documents MUST be able to decrypt with AES-128 in CBC mode.

Encryption based on [[W3C.REC-xmlenc-core-20021210](#)] does not assure integrity of the original document. There are known cryptographic attacks where someone who cannot decrypt a message can still change bits in a way where part or all the decrypted message makes sense but has a different meaning. Thus, Atom Processors that decrypt Atom Documents SHOULD check the integrity of the decrypted document by verifying the hash in the signature (if any) in the document, or by verifying a hash of the document within the document (if any).

## 5.3 Signing and Encrypting

When an Atom Document is to be both signed and encrypted, it is generally a good idea to first sign the document, then encrypt the signed document. This provides integrity to the base document while



encrypting all the information, including the identity of the entity that signed the document. Note that, if MACs are used for authentication, the order **MUST** be that the signed document is encrypted, and not the other way around.

## **6. Extending Atom**

### **6.1 Extensions From Non-Atom Vocabularies**

This specification describes Atom's XML markup vocabulary. Markup from other vocabularies ("foreign markup") can be used in an Atom Document. Note that the `atom:content` element is designed to support the inclusion of arbitrary foreign markup.

### **6.2 Extensions To the Atom Vocabulary**

The Atom namespace is reserved for future forwards-compatible revisions of Atom. Future versions of this specification could add new elements and attributes to the Atom markup vocabulary. Software written to conform to this version of the specification will not be able to process such markup correctly and, in fact, will not be able to distinguish it from markup error. For the purposes of this discussion, unrecognized markup from the Atom vocabulary will be considered "foreign markup".

### **6.3 Processing Foreign Markup**

Atom Processors which encounter foreign markup in a location that is legal according to this specification **MUST NOT** stop processing or signal an error. It might be the case that the Atom Processor is able to process the foreign markup correctly and does so. Otherwise, such markup is termed "unknown foreign markup".

When unknown foreign markup is encountered as a child of `atom:entry`, `atom:feed`, or a Person construct, Atom Processors **MAY** bypass the markup and any textual content and **MUST NOT** change their behavior as a result of the markup's presence.

When unknown foreign markup is encountered in a Text Construct or `atom:content` element, software **SHOULD** ignore the markup and process any text content of foreign elements as though the surrounding markup were not present.

### **6.4 Extension Elements**

Atom allows foreign markup anywhere in an Atom document, except where it is explicitly forbidden. Child elements of `atom:entry`, `atom:feed`, `atom:source`, and Person constructs are considered Metadata elements, and are described below. Child elements of Person constructs are considered to apply to the construct. The role of other foreign markup is undefined by this specification.



#### [6.4.1](#) Simple Extension Elements

A Simple Extension element MUST NOT have any attributes or child elements. The element MAY contain character data, or be empty. Simple Extension elements are not Language-Sensitive.

```
simpleExtensionElement =  
  element * - atom:* {  
    text  
  }
```

The element can be interpreted as a simple property (or name/value pair) of the parent element that encloses it. The pair consisting of the namespace-URI of the element and the local name of the element can be interpreted as the name of the property. The character data content of the element can be interpreted as the value of the property. If the element is empty, then the property value can be interpreted as an empty string.

#### [6.4.2](#) Structured Extension Elements

The root element of a Structured Extension element MUST have at least one attribute or child element. It MAY have attributes, it MAY contain well-formed XML content (including character data), or it MAY be empty. Structured Extension elements are Language-Sensitive.

```
structuredExtensionElement =  
  element * - atom:* {  
    (attribute * { text }+,  
     (text|anyElement)*)  
  | (attribute * { text }*,  
     (text?, anyElement+, (text|anyElement)*))  
  }
```

The structure of a Structured Extension element, including the order of its child elements, could be significant.

This specification does not provide an interpretation of a Structured Extension element. The syntax of the XML contained in the element, and an interpretation of how the element relates to its containing element is defined by the specification of the Atom extension.



## 7. IANA Considerations

An Atom Document, when serialized as XML 1.0, can be identified with the following media type:

MIME media type name: application

MIME subtype name: atom+xml

Mandatory parameters: None.

Optional parameters:

"charset": This parameter has identical semantics to the charset parameter of the "application/xml" media type as specified in [\[RFC3023\]](#).

Encoding considerations: Identical to those of "application/xml" as described in [\[RFC3023\]](#), [section 3.2](#).

Security considerations: As defined in this specification.

In addition, as this media type uses the "+xml" convention, it shares the same security considerations as described in [\[RFC3023\]](#), [section 10](#).

Interoperability considerations: There are no known interoperability issues.

Published specification: This specification.

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): As specified for "application/xml" in [\[RFC3023\]](#), [section 3.2](#).

File extension: .atom

Fragment identifiers: As specified for "application/xml" in [\[RFC3023\]](#), [section 5](#).

Base URI: As specified in [\[RFC3023\]](#), [section 6](#).





Macintosh File Type code: TEXT

Person and email address to contact for further information: Mark Nottingham <mnot@pobox.com>

Intended usage: COMMON

Author/Change controller: IESG

## **7.1 Registry of Link Relations**

This registry is maintained by IANA and initially contains five values: "alternate", "related", "self", "enclosure", and "via". New assignments are subject to IESG Approval, as outlined in [[RFC2434](#)]. Requests should be made by email to IANA, which will then forward the request to the IESG requesting approval. The request should use the following template:

- o Attribute Value: ( A value for the "rel" attribute that conforms to the syntax rule given in [Section 4.2.7.2](#) )
- o Description:
- o Expected display characteristics:
- o Security considerations:



## **8. Security Considerations**

### **8.1 HTML and XHTML Content**

Text constructs and atom:content allow the delivery of HTML and XHTML. Many elements in these languages are considered 'unsafe' in that they open clients to one or more types of attack. Implementers of software which processes Atom should carefully consider their handling of every type of element when processing incoming (X)HTML in Atom Documents. See the security sections of [\[RFC2854\]](#) and [\[HTML\]](#) for guidance.

Atom Processors should pay particular attention to the security of the IMG, SCRIPT, EMBED, OBJECT, FRAME, FRAMESET, IFRAME, META, and LINK elements, but other elements might also have negative security properties.

(X)HTML can either directly contain or indirectly reference executable content.

### **8.2 URIs**

Atom Processors handle URIs. See [Section 7 of \[RFC3986\]](#).

### **8.3 IRIs**

Atom Processors handle IRIs. See [Section 8 of \[RFC3987\]](#).

### **8.4 Spoofing**

Atom Processors should be aware of the potential for spoofing attacks where the attacker publishes an atom:entry with the atom:id value of an entry from another feed, perhaps with a falsified atom:source element duplicating the atom:id of the other feed. For example, an Atom Processor could suppress display of duplicate entries by displaying only one entry from a set of entries with identical atom:id values. In that situation, the Atom Processor might also take steps to determine whether the entries originated from the same publisher before considering them to be duplicates.

### **8.5 Encryption and Signing**

Atom Documents can be encrypted and signed using [\[W3C.REC-xmlenc-core-20021210\]](#) and [\[W3C.REC-xmlsig-core-20020212\]](#), respectively, and are subject to the security considerations implied by their use.

Digital signatures provide authentication, message integrity, and non-repudiation with proof of origin. Encryption provides data



confidentiality.

## **9. References**

### **9.1 Normative References**

- [HTML] Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification", W3C REC REC-html401-19991224, December 1999, <<http://www.w3.org/TR/1999/REC-html401-19991224>>.
- [MIMereg] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", work-in-progress ([draft-freed-media-type-reg-04](#)), April 2005.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2822] Resnick, P., "Internet Message Format", [RFC 2822](#), April 2001.
- [RFC2854] Connolly, D. and L. Masinter, "The 'text/html' Media Type", [RFC 2854](#), June 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", [RFC 3023](#), January 2001.
- [RFC3066] Alvestrand, H., "Tags for the Identification of Languages", [BCP 47](#), [RFC 3066](#), January 2001.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), July 2002.
- [RFC3548] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 3548](#), July 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), January 2005.
- [RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", [RFC 3987](#), January 2005.
- [W3C.REC-xml-20040204]  
Yergeau, F., Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-20040204>>.
- [W3C.REC-xml-c14n-20010315]





Boyer, J., "Canonical XML Version 1.0", W3C REC REC-xml-c14n-20010315, March 2001, <<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>>.

[W3C.REC-xml-exc-c14n-20020718]

Eastlake, D., Boyer, J., and J. Reagle, "Exclusive XML Canonicalization Version 1.0", W3C REC REC-xml-exc-c14n-20020718, July 2002, <<http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718>>.

[W3C.REC-xml-infoset-20040204]

Cowan, J. and R. Tobin, "XML Information Set (Second Edition)", W3C REC REC-xml-infoset-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-infoset-20040204>>.

[W3C.REC-xml-names-19990114]

Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, January 1999, <<http://www.w3.org/TR/1999/REC-xml-names-19990114>>.

[W3C.REC-xmlbase-20010627]

Marsh, J., "XML Base", W3C REC REC-xmlbase-20010627, June 2001, <<http://www.w3.org/TR/2001/REC-xmlbase-20010627>>.

[W3C.REC-xmlldsig-core-20020212]

Solo, D., Reagle, J., and D. Eastlake, "XML-Signature Syntax and Processing", W3C REC REC-xmlldsig-core-20020212, February 2002, <<http://www.w3.org/TR/2002/REC-xmlldsig-core-20020212>>.

[W3C.REC-xmlenc-core-20021210]

Reagle, J. and D. Eastlake, "XML Encryption Syntax and Processing", W3C REC REC-xmlenc-core-20021210, December 2002, <<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>>.

[XHTML]

Altheim, M., Boumphrey, F., McCarron, S., Dooley, S., Schnitzenbaumer, S., and T. Wugofski, "Modularization of XHTML[TM]", W3C REC REC-xhtml-modularization-20010410, April 2001, <<http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410>>.

## 9.2 Informative References

[ISO.8601.1988]

International Organization for Standardization, "Data



elements and interchange formats - Information interchange  
- Representation of dates and times", ISO Standard 8601,  
June 1988.

[RELAX-NG]

Clark, J., "RELAX NG Compact Syntax", December 2001,  
<[http://www.oasis-open.org/committees/relax-ng/  
compact-20021121.html](http://www.oasis-open.org/committees/relax-ng/compact-20021121.html)>.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an  
IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#),  
October 1998.

[W3C.NOTE-datetime-19980827]

Wolf, M. and C. Wicksteed, "Date and Time Formats", W3C  
NOTE NOTE-datetime-19980827, August 1998,  
<<http://www.w3.org/TR/1998/NOTE-datetime-19980827>>.

[W3C.REC-xmlschema-2-20041028]

Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes  
Second Edition", W3C REC REC-xmlschema-2-20041028,  
October 2004,  
<<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

Authors' Addresses

Mark Nottingham (editor)

Email: [mnot@pobox.com](mailto:mnot@pobox.com)

URI: <http://www.mnot.net/>

Robert Sayre (editor)

Email: [rfsayre@boswijck.com](mailto:rfsayre@boswijck.com)

URI: <http://boswijck.com>



## [Appendix A](#). Contributors

The following people contributed to preliminary drafts of this document: Tim Bray, Mark Pilgrim, and Sam Ruby. Norman Walsh provided the Relax NG schema. The content and concepts within are a product of the Atom community and the Atompub Working Group.

The Atompub Working Group has dozens of very active contributors who proposed ideas and wording for this document, including:

Danny Ayers, James Aylett, Roger Benningfield, Arve Bersvendsen, Tim Bray, Dan Brickley, Thomas Broyer, Robin Cover, Bill de hOra, Martin Duerst, Roy Fielding, Joe Gregorio, Bjoern Hoehrmann, Paul Hoffman, Anne van Kesteren, Brett Lindsley, Dare Obasanjo, David Orchard, Aristotle Pagaltzis, John Panzer, Graham Parks, Dave Pawson, Mark Pilgrim, David Powell, Julian Reschke, Phil Ringnalda, Antone Roundy, Sam Ruby, Eric Scheid, Brent Simmons, Henri Sivonen, Ray Slakinski, James Snell, Henry Story, Asbjorn Ulsberg, Walter Underwood, Norman Walsh, Dave Winer, and Bob Wyman.



## [Appendix B](#). RELAX NG Compact Schema

This appendix is informative.

The Relax NG schema explicitly excludes elements in the Atom namespace which are not defined in this revision of the specification. Requirements for Atom Processors encountering such markup are given in [Section 6.2](#) and [Section 6.3](#).

```
# -*- rnc -*-
# RELAX NG Compact Syntax Grammar for the
# Atom Format Specification Version 11

namespace atom = "http://www.w3.org/2005/Atom"
namespace xhtml = "http://www.w3.org/1999/xhtml"
namespace s = "http://www.ascc.net/xml/schematron"
namespace local = ""

start = atomFeed | atomEntry

# Common attributes

atomCommonAttributes =
  attribute xml:base { atomUri }?,
  attribute xml:lang { atomLanguageTag }?,
  undefinedAttribute*

# Text Constructs

atomPlainTextConstruct =
  atomCommonAttributes,
  attribute type { "text" | "html" }?,
  text

atomXHTMLTextConstruct =
  atomCommonAttributes,
  attribute type { "xhtml" },
  xhtmlDiv

atomTextConstruct = atomPlainTextConstruct | atomXHTMLTextConstruct

# Person Construct

atomPersonConstruct =
  atomCommonAttributes,
  (element atom:name { text }
  & element atom:uri { atomUri }?
  & element atom:email { atomEmailAddress }?)
```





```
    & extensionElement*)

# Date Construct

atomDateConstruct =
    atomCommonAttributes,
    xsd:dateTime

# atom:feed

atomFeed =
    [
        s:rule [
            context = "atom:feed"
            s:assert [
                test = "atom:author or not(atom:entry[not(atom:author)])"
                "An atom:feed must have an atom:author unless all "
                ~ "of its atom:entry children have an atom:author."
            ]
        ]
    ]
    element atom:feed {
        atomCommonAttributes,
        (atomAuthor*
        & atomCategory*
        & atomContributor*
        & atomGenerator?
        & atomIcon?
        & atomId
        & atomLink*
        & atomLogo?
        & atomRights?
        & atomSubtitle?
        & atomTitle
        & atomUpdated
        & extensionElement*),
        atomEntry*
    }

# atom:entry

atomEntry =
    [
        s:rule [
            context = "atom:entry"
            s:assert [
                test = "atom:link[@rel='alternate'] "
                ~ "or atom:link[not(@rel)] "
            ]
        ]
    ]
```



```
        ~ "or atom:content"
        "An atom:entry must have at least one atom:link element "
        ~ "with a rel attribute of 'alternate' "
        ~ "or an atom:content."
    ]
]
s:rule [
    context = "atom:entry"
    s:assert [
        test = "atom:author or "
        ~ "../atom:author or atom:source/atom:author"
        "An atom:entry must have an atom:author "
        ~ "if its feed does not."
    ]
]
]
element atom:entry {
    atomCommonAttributes,
    (atomAuthor*
    & atomCategory*
    & atomContent?
    & atomContributor*
    & atomId
    & atomLink*
    & atomPublished?
    & atomRights?
    & atomSource?
    & atomSummary?
    & atomTitle
    & atomUpdated
    & extensionElement*)
}

# atom:content

atomInlineTextContent =
    element atom:content {
        atomCommonAttributes,
        attribute type { "text" | "html" }?,
        (text)*
    }

atomInlineXHTMLContent =
    element atom:content {
        atomCommonAttributes,
        attribute type { "xhtml" },
        xhtmlDiv
    }
```



```
atomInlineOtherContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    (text|anyElement)*
  }

atomOutOfLineContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    attribute src { atomUri },
    empty
  }

atomContent = atomInlineTextContent
| atomInlineXHTMLContent
| atomInlineOtherContent
| atomOutOfLineContent

# atom:author

atomAuthor = element atom:author { atomPersonConstruct }

# atom:category

atomCategory =
  element atom:category {
    atomCommonAttributes,
    attribute term { text },
    attribute scheme { atomUri }?,
    attribute label { text }?,
    undefinedContent
  }

# atom:contributor

atomContributor = element atom:contributor { atomPersonConstruct }

# atom:generator

atomGenerator = element atom:generator {
  atomCommonAttributes,
  attribute uri { atomUri }?,
  attribute version { text }?,
  text
}
```



# atom:icon

```
atomIcon = element atom:icon {  
    atomCommonAttributes,  
    (atomUri)  
}
```

# atom:id

```
atomId = element atom:id {  
    atomCommonAttributes,  
    (atomUri)  
}
```

# atom:logo

```
atomLogo = element atom:logo {  
    atomCommonAttributes,  
    (atomUri)  
}
```

# atom:link

```
atomLink =  
    element atom:link {  
        atomCommonAttributes,  
        attribute href { atomUri },  
        attribute rel { atomNCName | atomUri }?,  
        attribute type { atomMediaType }?,  
        attribute hreflang { atomLanguageTag }?,  
        attribute title { text }?,  
        attribute length { text }?,  
        undefinedContent  
    }
```

# atom:published

```
atomPublished = element atom:published { atomDateConstruct }
```

# atom:rights

```
atomRights = element atom:rights { atomTextConstruct }
```

# atom:source

```
atomSource =  
    element atom:source {  
        atomCommonAttributes,
```





```
(atomAuthor*
  & atomCategory*
  & atomContributor*
  & atomGenerator?
  & atomIcon?
  & atomId?
  & atomLink*
  & atomLogo?
  & atomRights?
  & atomSubtitle?
  & atomTitle?
  & atomUpdated?
  & extensionElement*)
}

# atom:subtitle

atomSubtitle = element atom:subtitle { atomTextConstruct }

# atom:summary

atomSummary = element atom:summary { atomTextConstruct }

# atom:title

atomTitle = element atom:title { atomTextConstruct }

# atom:updated

atomUpdated = element atom:updated { atomDateConstruct }

# Low-level simple types

atomNCName = xsd:string { minLength = "1" pattern = "[^:]*" }

# Whatever a media type is, it contains at least one slash
atomMediaType = xsd:string { pattern = ".+/.+" }

# As defined in RFC 3066
atomLanguageTag = xsd:string {
  pattern = "[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*"
}

# Unconstrained; it's not entirely clear how IRI fit into
# xsd:anyURI so let's not try to constrain it here
atomUri = text

# Whatever an email address is, it contains at least one @
```



```
atomEmailAddress = xsd:string { pattern = ".+@.+" }

# Simple Extension

simpleExtensionElement =
  element * - atom:* {
    text
  }

# Structured Extension

structuredExtensionElement =
  element * - atom:* {
    (attribute * { text }+,
     (text|anyElement)*)
  | (attribute * { text }*,
     (text?, anyElement+, (text|anyElement)*))
  }

# Other Extensibility

extensionElement =
  simpleExtensionElement | structuredExtensionElement

undefinedAttribute =
  attribute * - (xml:base | xml:lang | local:*) { text }

undefinedContent = (text|anyForeignElement)*

anyElement =
  element * {
    (attribute * { text }
     | text
     | anyElement)*
  }

anyForeignElement =
  element * - atom:* {
    (attribute * { text }
     | text
     | anyElement)*
  }

# XHTML

anyXHTML = element xhtml:* {
  (attribute * { text }
   | text
```



```
        | anyXHTML)*
    }

    xhtmlDiv = element xhtml:div {
        (attribute * { text }
        | text
        | anyXHTML)*
    }

    # EOF
```

## [Appendix C](#). Change Log

[[anchor70: This section should be removed before final publication.]]

-10: capitalize "Atom Document" consistently.

fix atom:feed/atom:logo

fix link hreflang/alternate in atom:feed

Add more acknowledgements

Expand security section

Clarify IRI processing

-09: Changed atom:copyright to atom:rights.

Clarify atom:source, also reflect changes to atom:feed

Change 'minimal' to brief (PaceBriefExample).

Add text about "Atom 1.0" (PaceAtom10).

Remove SHOULD about content@src.

feed/id now required.

rework xml:base text

rework xml:lang terms and requirements

rework escaped markup explanations

change atom:image to atom:logo

Add example Date Constructs

Clarify that atom:link and atom:category MUST be empty.

Make XHTML definitions consistent

atom:icon editorial fix-- capitalize SHOULD.



-08: Remove BNF

complete rather than collected schema

Remove a couple introductory sentences

update MIME references

Many editorial adjustments

-07: Change atom:source-feed to atom:source.

Add ABNF reference

Many editorial tweaks

Rework extensibility

Adjust page breaks in txt version

-06: Move Identity Construct into atom:id (only place it's used)

atom:id values must be unique within a feed.

restore atom:copyright definition mistakenly dropped during alphabetizing.

Remove atom:head, add atom:source-feed, and "Extension Construct" text in an effort to accurately reflect WG consensus on data model and extensibility, acknowledging two opinions in favor of atom:head.

Note @hreflang issue.

Add comment on atom:entry/atom:summary requirements.

Rework atom:id text. The dereferencing section didn't talk about dereferencing.

Remove protocol reference.

Alphabetize where appropriate (PaceOrderSpecAlphabetically).

Add mI language (PaceExtendingAtom).





Add atom:icon and atom:image (PaceImageAndIcon).

Change atom:tagline to atom:subtitle

Add inline XHTML language (PaceXHTMLNamespaceDiv).

Change "TEXT" etc, to lowercase

Change example id IRI to urn:uuid:...

Add rel="self" (PaceFeedLink).

Add Feed State text (PaceNoFeedState).

Move to IRIs (PaceIRI).

Add rel="via" (PaceLinkRelVia).

Add rel="enclosure" (PaceEnclosuresAndPix).

Remove info and host (PaceRemoveInfoAndHost)

Clarify order of entries (PaceEntryOrder).

Remove version attribute (PaceRemoveVersionAttr).

Date format roundup (PaceDatesXSD).

Remove Service construct and elements.

fix atom:contributor cardinality typo

Removed motivation/design principles note; if we haven't come up with them by now...

Put conformance text into notational conventions.

Removed instances of 'software'; too specific.

Added refs to HTML and XHTML.

Updated ref to Infoset.

Various editorial tweaks.



Fix [RFC 3023](#) refs in IANA section

Adjust head/link requirement

fix @version typos

-05: Add RNC from N. Walsh.

Re-organize element definitions.

Lift the prohibition on other types of DSig and encryption.

Remove text on "indiscriminate use" of DSig and XMLEnc.

-04: Update URI terms for 2396bis.

Add Category construct (PaceCategoryRevised).

Insert paranoid XHTML interpretation guidelines.

Adjust atom:copyright, per chairs' instruction.

Add atom:host as child element of atom:entry, per chairs' direction (PacePersonConstruct).

Add link/content co-constraint (PaceContentOrLink).

Remove atom:origin as a side effect of adding atom:head to atom:entry (PaceHeadInEntry).

Add optional length attribute to atom:link (PaceLinkRelated).

Add Link registry to Link Construct, IANA Considerations placeholder (PaceFieldingLinks).

Change definition of atom:updated (PaceUpdatedDefinition).

-03: Move definition of Link @rel to format spec, restrict acceptable values (PaceMoveLinkElement, PaceLinkAttrDefaults).

Add Service Construct, head/post, head/introspection, entry/edit (PaceServiceElement).

Add Text Construct, entry/content (PaceReformedContent3).



Add entry/published (PaceDatePublished).

Adjust definition of Identity Construct per chairs' direction to "fix it."

Add Sayre to editors.

- 02: Removed entry/modified, entry/issued, entry/created; added entry/updated (PaceDateUpdated).

Changed date construct from W3C date-time to [RFC3339](#) (PaceDateUpdated).

Feed links to HTML pages should be reflected back (PaceLinkReflection).

Added Identity construct (PaceIdConstruct).

Changed feed/id and entry/id to be Identity constructs (PaceIdConstruct).

Changed entry/origin's content so that it's the same as the feed's id, rather than its link/@rel="alternate" (PaceIdConstruct).

Added "Securing Atom Documents" (PaceDigitalSignatures).

- 01: Constrained omission of "Information Item" to just elements and attributes.

Clarified xml:lang inheritance.

Removed entry- and feed-specific language about xml:lang (covered by general discussion of xml:lang)

Changed xml:lang to reference XML for normative requirements.

Changed "... MUST be a string" to "... is unstructured text."

Removed language about DOCTYPEs, PIs, Comments, Entities.

Changed atom:url to atom:uri, @url to @uri

Introduced atom:head

Introduced "Atom Feed Document" and "Atom Entry Document".



Removed requirement for all elements and attributes to be namespace-qualified; now children of selective elements

Added extensibility to Person constructs.

Removed requirement for media types to be registered (non-registered media types are legal)

Added atom:origin (PaceEntryOrigin)

Added requirement for entry/id to be present and a URI (PaceEntryIdRequired).

Clarified approach to Comments, PIs and well-formedness, as per [RFC3470](#).

Referenced escaping algorithm in XML.

Assorted editorial nits and cleanup, refactoring

-00: Initial IETF Internet-Draft submission.

Added optional version attribute to entry (PaceEntryElementNeedsVersionAttribute).

Added hreflang attribute (PaceHrefLang).

Clarified inheritance of copyright element (PaceItemCopyright).

Added xml:lang to entries (PaceItemLang).

Tweaked InfoSet-related language (PaceNoInfoSet).

Clarified lack of structure in version attribute (PaceVersionAsText).

Changed approach to XML Base (PaceXmlBaseEverywhere).

Added XML Base processing to atom:id (PaceXmlBaseId).

Various editorial cleanup and adjustments for IETF publication.





## Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

## Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

## Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

